Redictado Taller de Programación 2021

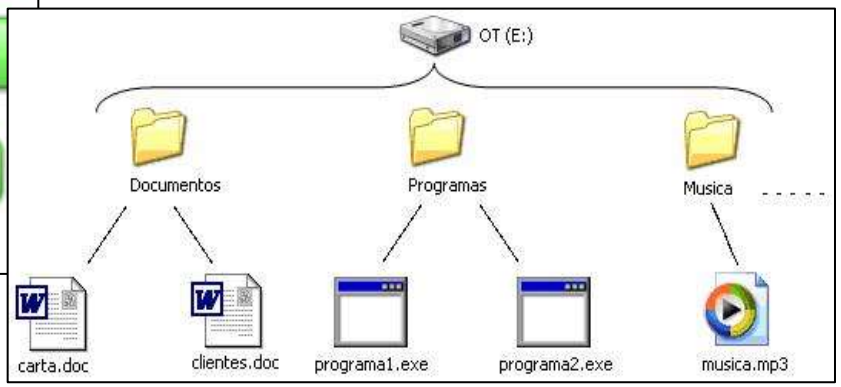
CLASE 2

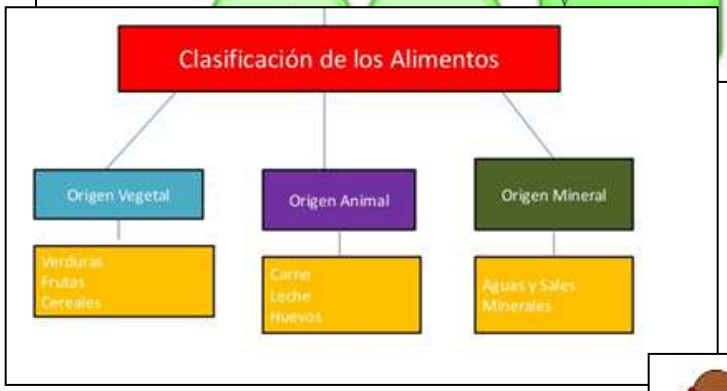
Recursión - Árboles

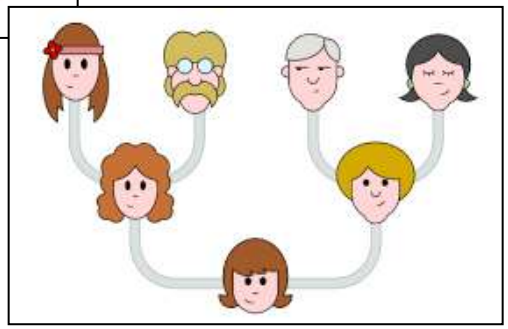
Temas de la clase

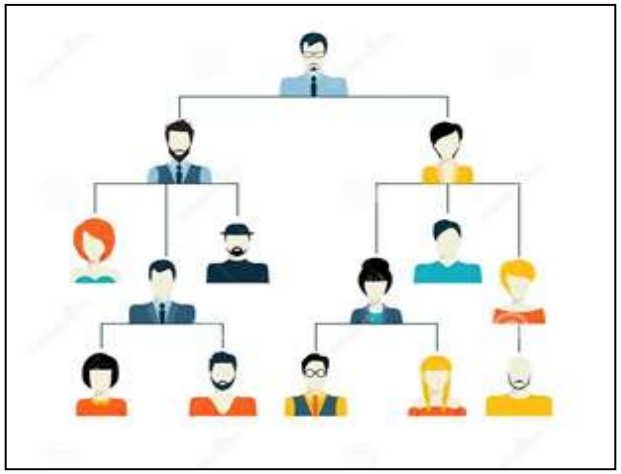
- Caso de uso de Recursión: Árbol Binario
- Árboles Binarios. Definición y características.
- Operaciones con Árboles Binarios de Búsqueda


ANIMALES
EQUINODERMOS
MAMÍFEROS
VERTEBRADOS
INVERTEBRADOS
ARTRÓPODOS
PECES
AVES
PORÍFEROS
MOLUSCOS
GUSANOS


OT (E:)
Documentos
Programas
Musica
carta.doc
clientes.doc
programa1.exe
programa2.exe
musica.mp3


Clasificación de los Alimentos
Origen Vegetal
Origen Animal
Origen Mineral
Verduras
Frutas
Cereales
Carne
Leche
Huevos
Aguas y Sales
Minerales

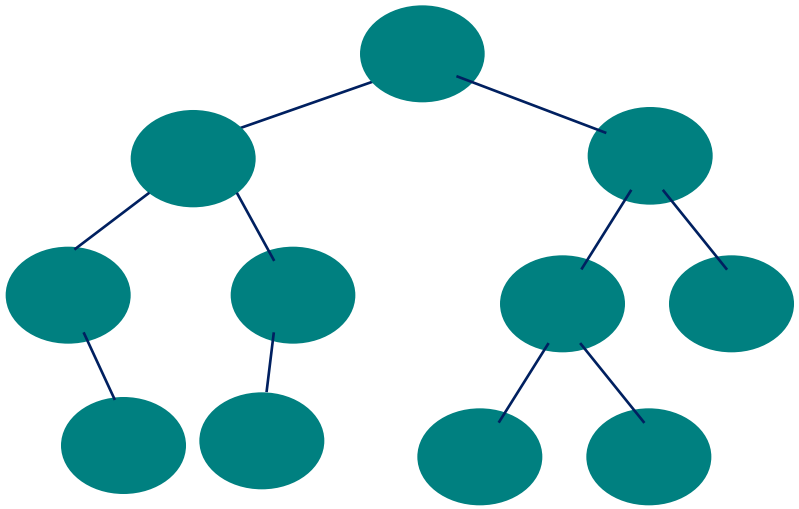




Árbol Binario – Definición y Características

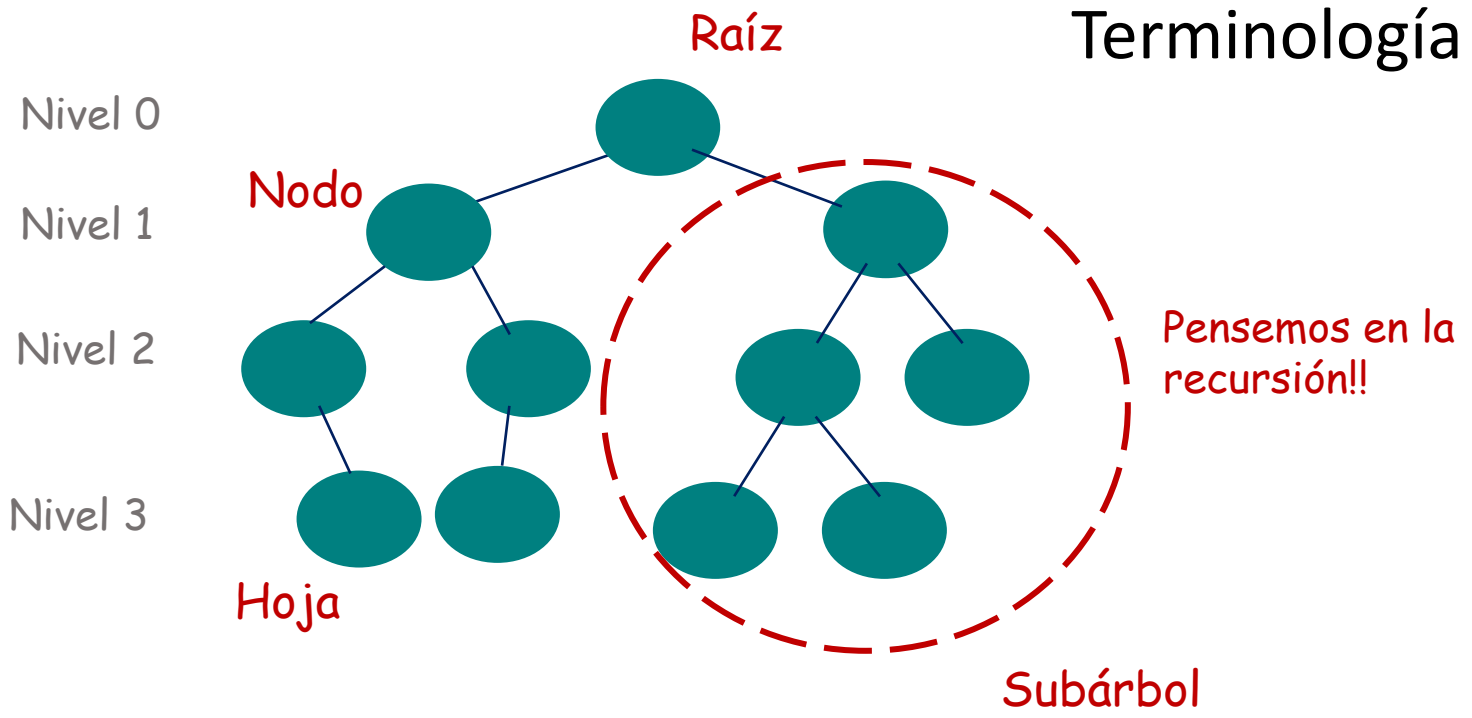
Un **árbol binario** es una estructura de datos con las siguientes características:

1. Homogénea: todos los elementos son del mismo tipo
2. Dinámica: puede aumentar o disminuir su tamaño durante la ejecución del programa
3. No lineal: cada elemento puede tener 0, 1, o 2 sucesores
4. Acceso Secuencial



Árbol Binario – Definición y Características

- Cada elemento del árbol se relaciona con cero, 1 o 2 elementos (hijos).
- Si el árbol no está vacío, hay un único elemento (raíz) y que no tiene padre (predecesor).
- Todo otro elemento del árbol posee un único padre y es un descendiente de la raíz.

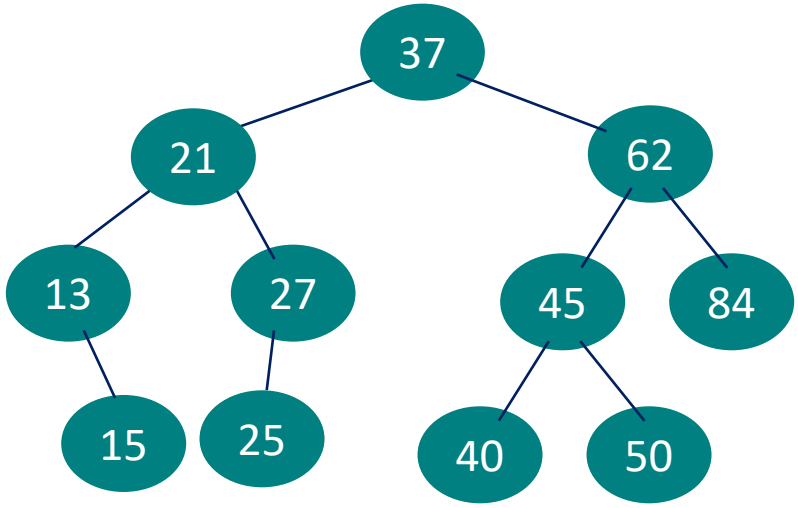


Árbol Binario de Búsqueda (ABB)

Cada nodo tiene un valor que

- Es mayor que el valor de todos los nodos del subárbol izquierdo
- Es menor que el valor de todos los nodos del subárbol derecho

Utilidad más importante ➔ Búsquedas: el tiempo medio es O(log n))



Árbol Binario - Representación

```
Type

  elemento = tipoElemento;

  arbol = ^nodo;

  nodo = record
           elem: elemento;
           hijoIzq: arbol;
           hijoDer: arbol;
         end;
```

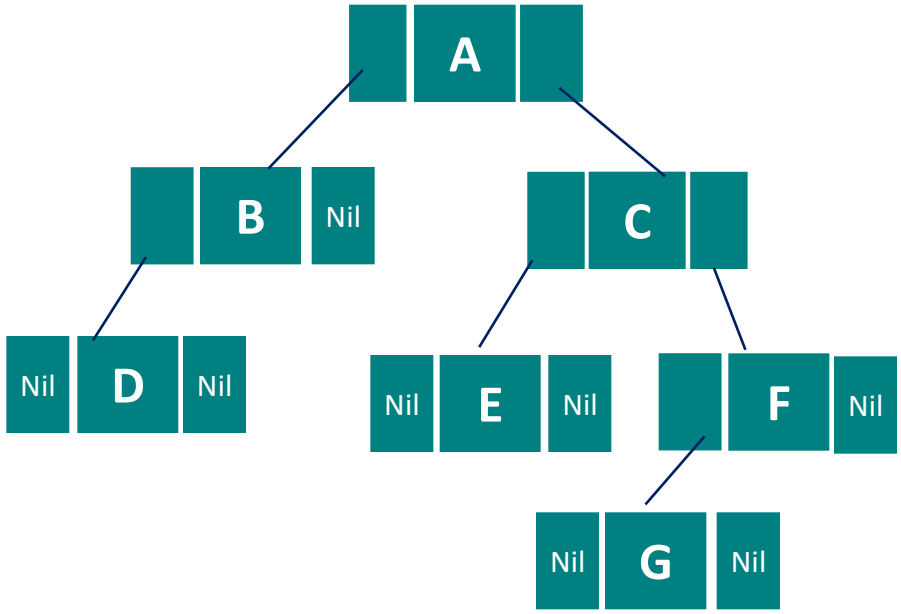


ABB – Operación Insertar un dato

Consideraciones:

- Al principio el árbol esta vacío (puntero a raíz **a** es nil)
- Siempre se inserta a nivel hoja (respetando criterio orden)
- Supongamos que se lee: 20 7 36 1 4 23

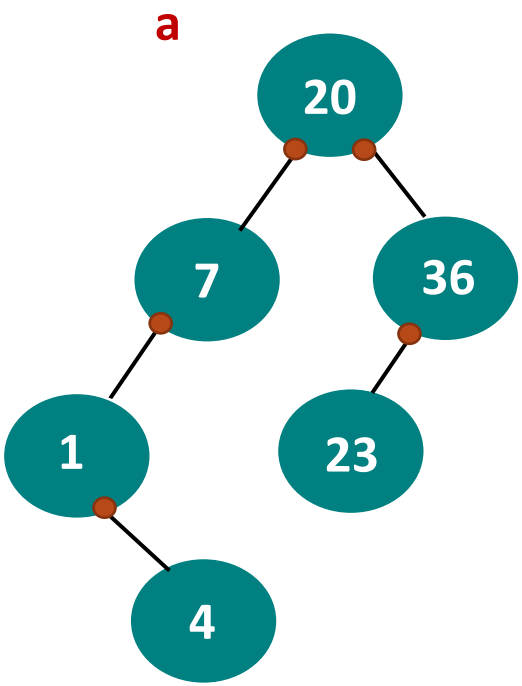


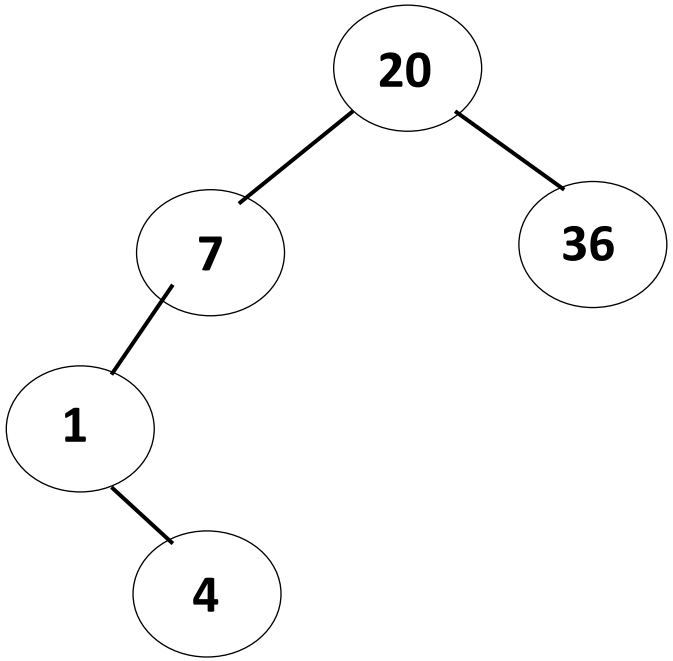
ABB – Operación Insertar un dato - PSEUDOCÓDIGO

```
insertar (arbol, dato)
  si arbol es nil
    creo nodo_nuevo y pongo el dato y los hijos en nil
    arbol := nodo_nuevo
  sino
    si el dato en árbol es > dato
      insertar(hijo_izquierdo_del_árbol, dato);
    sino
      insertar(hijo_derecho_del_árbol, dato);
```

ABB – Operación Insertar un dato - CÓDIGO

```
Type
  arbol= ^nodoA;
  nodoA = Record
    dato: integer;
    HI: arbol;
    HD: arbol;
  End;

procedure Insertar(var a: arbol; dato: integer);
begin
   if (a = nil) then begin
          new(a);
          a^.dato:= dato;
          a^.HI:= nil;
          a^.HD:= nil;
   end
   else
      if (a^.dato > dato) then
         Insertar(a^.HI, dato)
      else
         Insertar (a^.HD, dato);
end;
```



¿Cómo se inserta el 10?

¿Cómo se inserta el 1?

¿Cómo evitaría insertar repetidos?

¿Cómo contaría las apariciones de cada valor?

ABB – Recorridos

Los distintos recorridos permiten desplazarse a través de todos los nodos del árbol de tal forma que cada nodo sea visitado una y solo una vez.

Existen varios métodos que se diferencian en el orden que se visitan los nodos:

- **Recorrido En – Orden** (subárbol izquierdo – raíz – subárbol derecho)
- **Recorrido Pre – Orden** (raíz - subárbol izquierdo – subárbol derecho)
- **Recorrido Post – Orden** (subárbol izquierdo – subárbol derecho - raíz)

```
procedure enorden(a:arbol);
begin
  if (a <> nil) then begin
      enorden (a^.HI);
      write (a^.dato);
      enorden (a^.HD);
  end;
end;
```

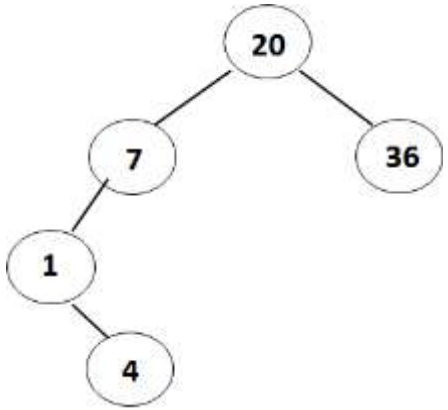
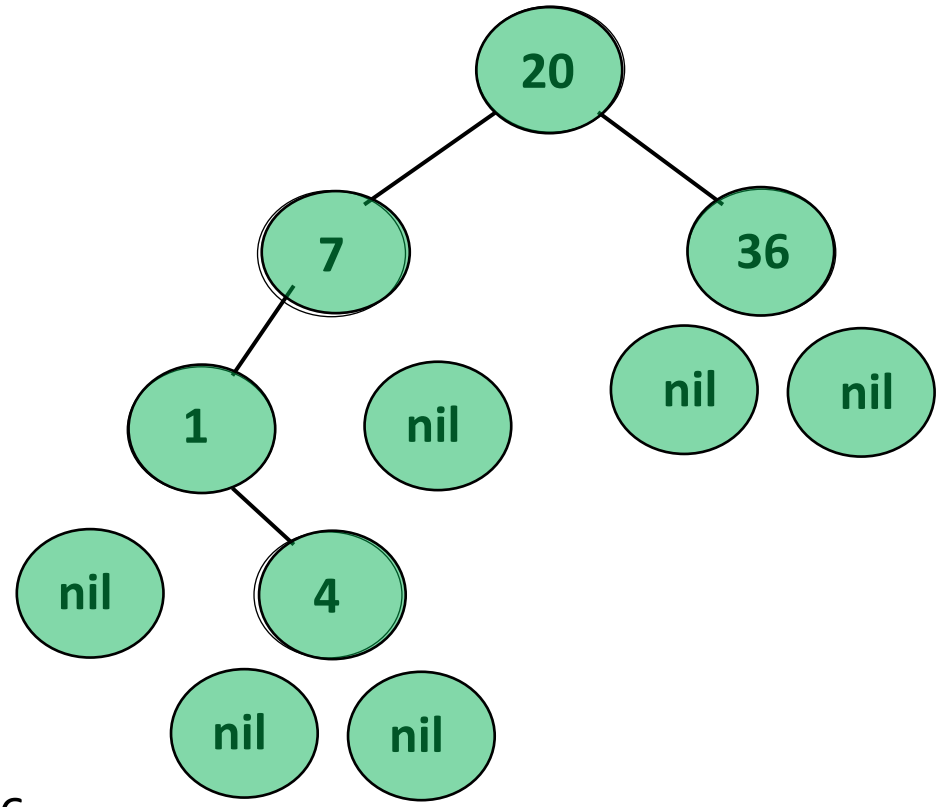


ABB – Recorridos – En Orden

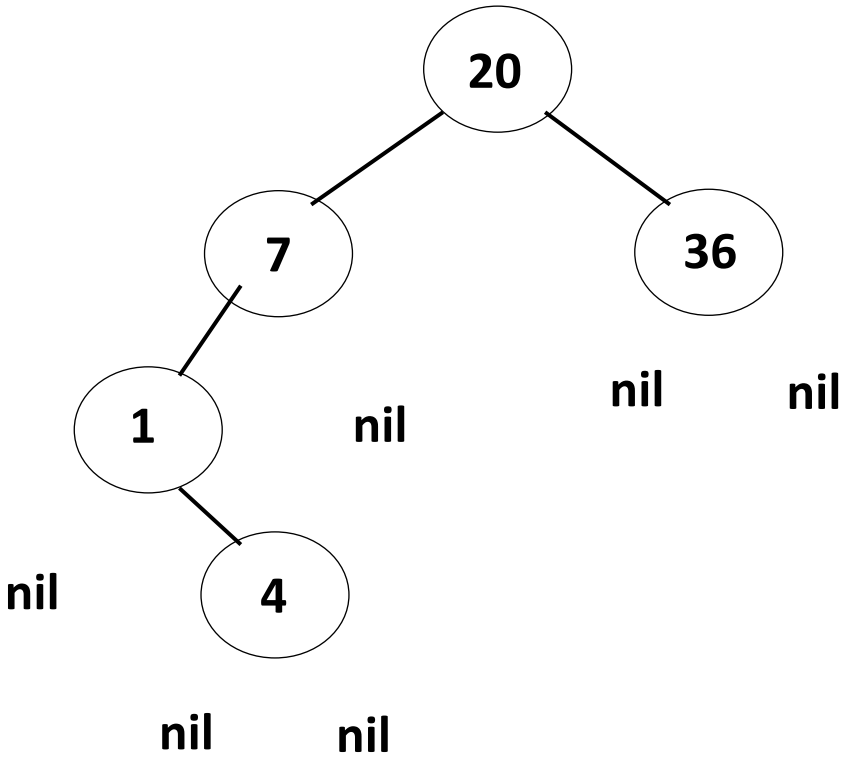
```
procedure enorden(a:arbol);
begin
  if (a <> nil) then begin
1.  enorden (a^.HI);
2.  write (a^.dato);
3.  enorden (a^.HD);
  end;
end;
```



Salida: 1 4 7 20 36

ABB – Recorridos – Pre Orden

```
procedure preorden(a:arbol);
begin
  if (a <> nil) then begin
  1. write (a^.dato);
  2. preorden(a^.HI);
  3. preorden(a^.HD);
  end;
end;
```

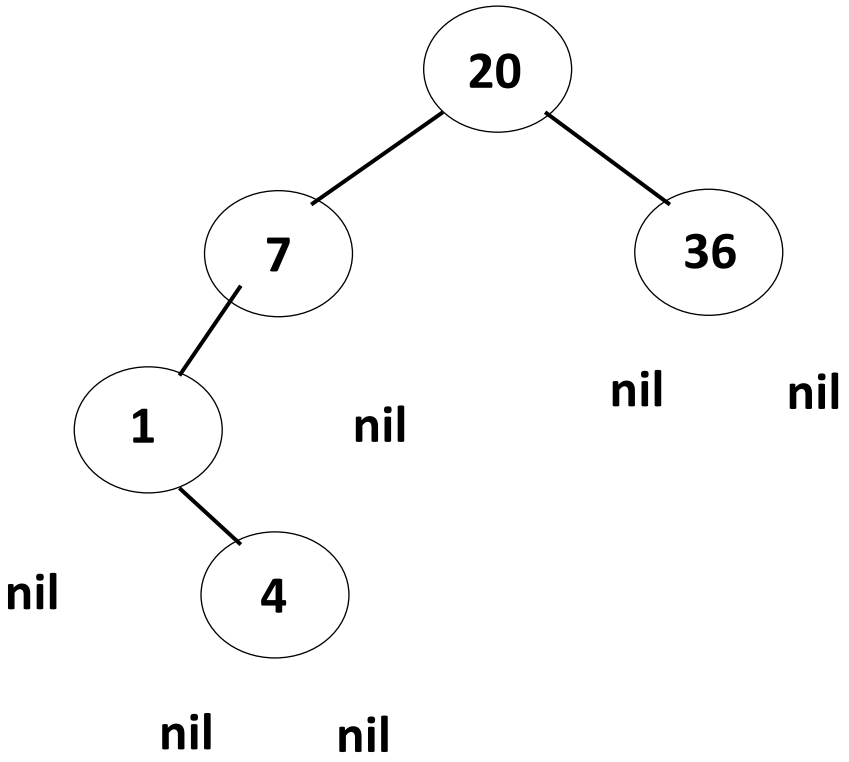


Salida:

¿Qué imprime?

ABB – Recorridos – Post Orden

```
procedure postorden(a:arbol);
begin
  if (a <> nil) then begin
  1. postorden(a^.HI);
  2. postorden(a^.HD);
  3. write (a^.dato);
  end;
end;
```



Salida:

¿Qué imprime?

ABB – Buscar

Permite buscar un elemento considerando el criterio de orden del árbol, retornando su puntero o nil si no existe.

¿Cómo busco el 4?

¿Cómo busco el 23?

Comparamos el valor a buscar con la raíz y si no coincide, continuamos nuestra búsqueda por el subárbol derecho en caso de que sea mayor, o lo hacemos por el izquierdo en caso que sea menor.

La búsqueda finaliza al encontrar el valor o llegar a nil (no existía el valor).

La búsqueda en un ABB es en general, más rápida que la búsqueda secuencial en un vector o una lista.

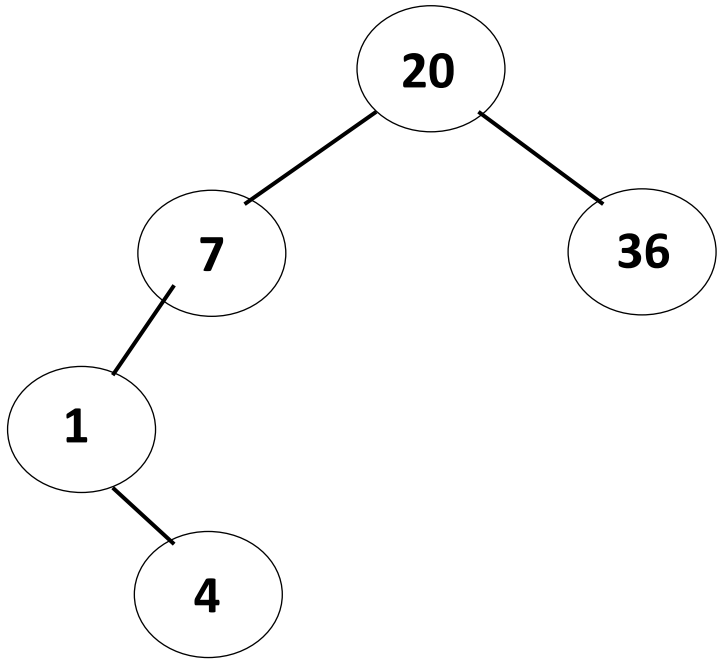


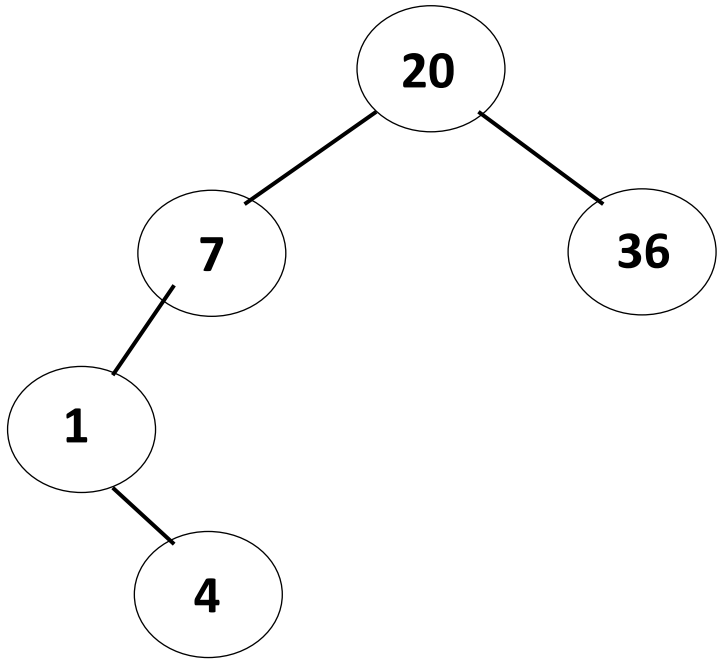
ABB – Buscar

Permite buscar un elemento considerando el criterio de orden del árbol, retornando su puntero o nil si no existe.

¿Cómo busco el 4?

¿Cómo busco el 23?

```
function Buscar (a:arbol; dato: integer):arbol;
begin
  if (a=nil) then
     Buscar:=nil
  else
     if (dato= a^.dato) then Buscar:=a
     else
         if (dato < a^.dato) then
            Buscar:=Buscar(a^.HI ,dato)
         else
            Buscar:=Buscar(a^.HD ,dato);
end;
```



¿Cuál/es son los casos base de las operaciones vistas?

Actividades en Máquina

ACTIVIDAD

Realice un programa que contenga:

a) Un módulo que cargue un ABB con valores enteros generados al azar (finaliza con 0). De cada "valor" entero el ABB debe almacenar la cantidad de veces que apareció en la secuencia. El ABB se ordena por "valor".

b) Un módulo que reciba el ABB y devuelva la cantidad de valores que aparecieron más de X veces (X se recibe).

c) Un módulo que reciba el ABB y devuelva el "valor" mínimo almacenado.

d) Un módulo que reciba el ABB y un "valor", y devuelva el puntero su nodo o nil en caso de no existir.